Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 6 (2023)

Enhancing Service Efficiency and Ensuring Privacy in Distributed Computing Environments through a MapReduce-Based Framework.

Hemant Mathur[1], Dr. Reema Ajmera[2]
Reaserch Scholar[1], Professor & Dean[2]
Computer Science & Applications[12]
Nirwan University, Japur, India[12]

Abstract: This research focuses on enhancing service efficiency and privacy assurance in distributed computing environments, specifically within the Collaborative Big Data Computation and Multiple Public Clouds (CBDC-MPC) paradigm. The study explores authentication mechanisms within CBDC-MPC, emphasizing their implications for risks, security provisioning, and authentication. A crucial connection is established with the MapReduce paradigm to align the proposed authentication framework with service efficiency goals. The research includes a comparative analysis of entity authentication techniques within CBDC-MPC, aiming to optimize security levels while minimizing overhead costs. The overall narrative positions authentication within CBDC-MPC as an integral part of a larger framework, where service efficiency, privacy, and authentication seamlessly converge.

Key words: Distributed computing, Collaborative Big Data Computation (CBDC), Multiple Public Clouds (MPC), Authentication mechanisms, Service efficiency, Privacy assurance, MapReduce paradigm.

1. Introduction:

Big Data computing is a new kind of computing that analyses and extracts useful information from massive amounts of data or data that are too complicated for regular data processing systems to handle effectively. Many industries, including healthcare, agriculture, and environmental sustainability, have adopted big data computing. This includes a rise in Big Data processing and inter-organizational data sharing, where cooperative organisations worked together to analyse common information [1]. CBDC is often carried out utilising distributed computing services for efficiency reasons. These services are often housed in public clouds. It is expected that cloud services and distributed computing services are offered by various third-party service providers in keeping with a trend towards computing as a utility. Using a Collaborative Big Data Computation and Multiple Public Cloud (CDBC-MPC) platform for collaborative big data computation raises a number of security issues [2].

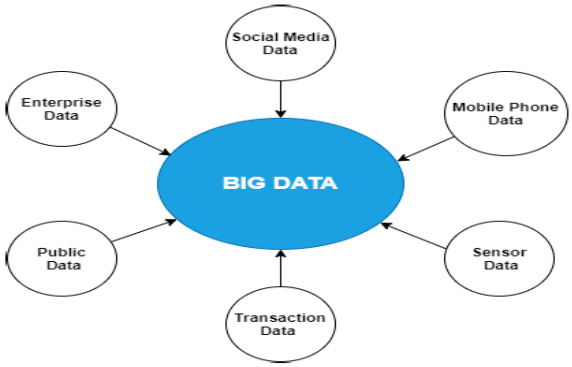


Fig. 1: The Big Data World

This paper integrates two critical aspects of data security CBDC-MPC and Big Data Outsourcing. This chapter unfolds a comprehensive narrative, highlighting the construction of a General Use Case Model for CBDC-MPC, the MapReduce-based Innovative Entity Authentication (MDA) framework, and the intricacies of crafting robust security architecture for outsourcing big data to the cloud [3].

2. Experiment Implementation

Experimental assessments were used to assess the MIEA and CPDA frameworks' performances. On real-system testbeds using mock-up and real-world datasets, two sets of experiments were run with various parameter configurations. assessment metrics were established, assessment techniques were developed, and parameters for cryptographic building blocks were explored for experiment settings. Then, using the Botan cryptographic library and the C++ and Python programming languages, the components (i.e., authentication techniques and protocols) of the MIEA and CPDA frameworks were put into practice [4].

3. Data Outsourcing

3.1 Defining Components and Interactions

In this section, the essential components and interactions within the CBDC-MPC ecosystem are defined. From master nodes to worker nodes, data storage, and communication patterns, the model meticulously delineates the entities involved and their collaborative relationships during big data computation [5].

3.2 Integrating MapReduce Paradigm

The incorporation of the MapReduce paradigm into the CBDC-MPC use case model is explored. This integration provides a structured framework for understanding the collaborative nature of big data processing and lays the groundwork for subsequent security considerations.

3.3 Identifying Potential Threats

A thorough analysis of potential threats within the CBDC-MPC environment is conducted. This includes external and internal threats, data breaches, unauthorized access, and other vulnerabilities that may compromise the security and integrity of collaborative big data computation[6].

4. Contextual Threat Classifications

Building on the identified threats, this sub-section introduces contextual threat classifications. These classifications categorize threats based on their origin, impact, and potential vectors, providing a structured approach to developing countermeasures [7].

4.1 Requirement Specifications

4.1.1 Security and Efficiency Balance

This section outlines the specific requirements necessary to address the identified threats while maintaining the efficiency goals of CBDC-MPC. The delicate balance between security and efficiency becomes a guiding principle for the subsequent development of security frameworks.

4.2.3.2 Scalability and Flexibility

In response to the collaborative and dynamic nature of CBDC-MPC, scalability and flexibility requirements are defined. The model considers the varying scales of data processing and the need for adaptive security measures in diverse scenarios [8].

4.2.4 Summary of MDA Framework

4.2.4.1 Core Principles

A concise summary of the MapReduce-based Innovative Entity Authentication (MDA) framework is presented, highlighting its core principles. MDA's alignment with the MapReduce paradigm and its innovative approach to entity authentication within CBDC-MPC are succinctly outlined [9].

4.2.4.2 Contributions to Security

This sub-section emphasizes the contributions of the MDA framework to security within CBDC-MPC. By seamlessly integrating entity authentication with the collaborative processing model, MDA becomes a key component in fortifying the overall security posture.

4.2.4.3 Big data outsourcing from data owners

(i) Identification of an efficient lightweight algorithm suitable for big data applications.

(ii) Identification of appropriate compression methods applicable to big data for the effective utilization of the cloud.

(iii) Identification of an efficient clustering method to facilitate easy access and sharing with data users.

(iv) Development of a mechanism for data sharing with the consent of the data owner [10].

Big data outsourcing from data owners to cloud is in three phases as follows:

1) Authentication

2) Compression

3) Encryption

4.3 Authentication

Authentication is the method of verifying a user's identity. It involves associating an incoming request with a specific set of identifying credentials. These credentials are compared with the information stored in a dataset containing the approved user's data within a database or an authentication server. The authentication process typically initiates at the beginning of the application, ensuring that no other code is permitted to proceed until user identity is confirmed [11].

4.3.1 Outsourcing Big Data from Data Owners

When data owners intend to outsource data to the Cloud Server (CS), they must first register their identities with the Trust Center (TC). This process involves three key steps:

Registration: The data owner registers with the TC, providing the following identities: Email ID, User ID, Password, Current Timestamp, and Secure ID. User ID and Password undergo hashing before being submitted to TC for registration. Post-registration, TC generates a hash value for the information provided by the data owner using SHA3-384, storing it in the database. Refer to Fig. 3 for details of the registration process.

Login: Upon logging into the CS, the data owner must furnish the following information: Email ID, User ID, Password, Current Timestamp, and Secure ID. Subsequently, they await a successful authentication response from CS.

Authentication: During login, the information provided by the data owner is hashed and compared with the database. The step-by-step description of user authentication is illustrated in Fig. 4.

Upon successful authentication with the TC, the data owner requests a private key for data encryption. TC generates a private key based on the sensitivity level specified by the data owner. Three sensitivity levels are defined: (i) Non-sensitive, (ii) Sensitive, and (iii) Most-sensitive [12].

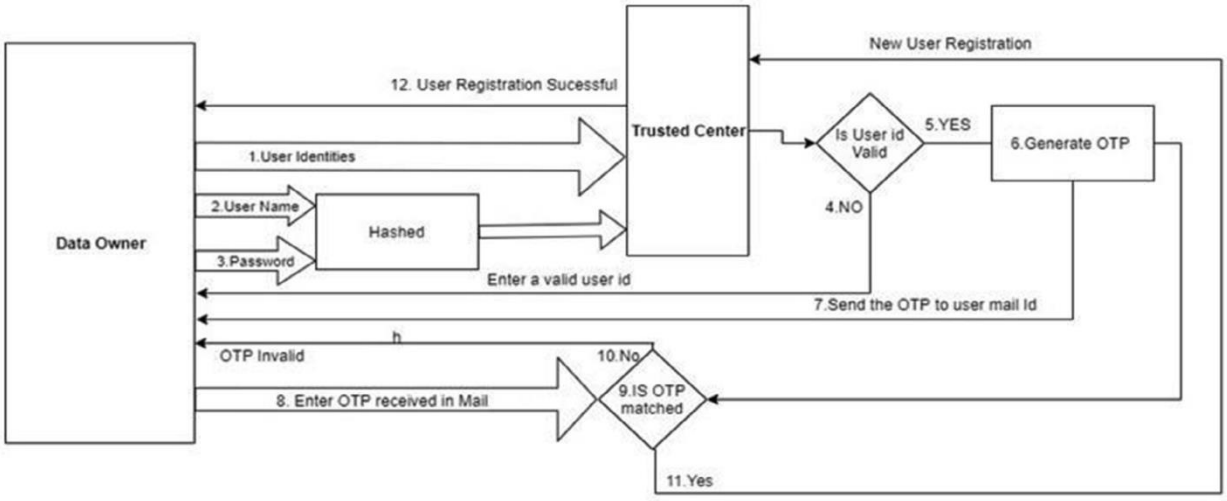


Fig. 2: Data User Registration with Trusted Center.

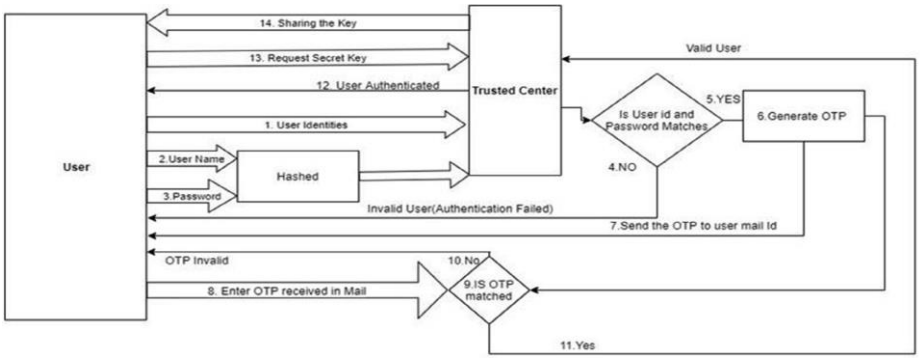


Fig. 3: User Authentication with Trusted Center

4.3.2 SHA3 Hashing AlgorithmSHA3, a secure hashing algorithm, offers four hashing functions: SHA3-224, SHA3-256, SHA3-384, and SHA3-512. It also incorporates two Extendable Output Functions (XOFs): SHAKE-128 and SHAKE-256. The SHA3 hashing algorithm is employed for message authentication during the registration processes of both data owners (DO) and data users (DU). It is based on the Keccak algorithm with Sponge Construction.

Pseudocode for SHA-3 Hashing Algorithm

Step 1) Begin

Step 2) SHA3:= PROC(M::STRING, MT::Name:=TEXT)

Step 3) Local N;M; L;

Step 4) If Type (PROCname,"INDEXED")

Step 5) Then N:= OP ('PROCname')

Step 6) Else

Step 7) Error 'Output length is not specified'

Step 8) End If;

Step 9) If Not N in Output (224, 256, 384, 512) Then

Step 10) Error 'Not a Valid Output Length',N

Step 11) End If;

Step 12) M:= MessagetoBytes (M, MT);

Step 13) L:= Keccak (M, 1600, 1600–2.N, N, Hash);

Step 14) BytetoHexStringðLÞ;

Step 15) EndPROC

In the process of hashing, the input involves padding functions, wherein messages are provided as a list of integers or bytes within the range of 0 to 255. This process considers parameters like Domain and Bit Rate. Under the domain, it takes into account Hash, XOF, and KEC, necessitating various paddings and implementing Domain Separation by distinguishing input and corresponding to a hash function. Subsequently, an output is generated for this padding function, resulting in an array containing padded message blocks, with each block consisting of a list of integers. This entire procedure is applied to hash user authentication securely, both at the Trust Center (TC) and the Cloud Server (CS). The term "PROC" denotes the procedure, "N" represents the output length in bits (224, 256, 384, and 512), "M" stands for the message, and "MT" indicates the message type. The complete procedure is executed for hashing data owner (DO) and data user (DU) information. Figure 4 illustrates the duration required for key generation and authentication processes.

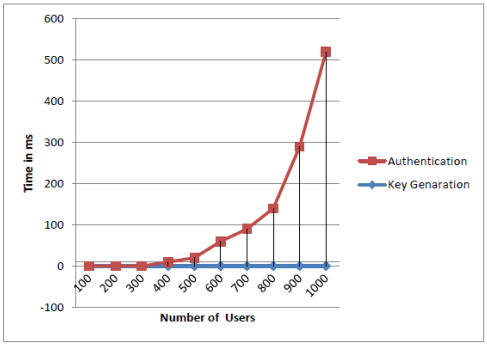


Fig. 4: Time taken for Key Generation and Authentication

4.4. Data Compression

Compression refers to the process of reducing data size without compromising information integrity, offering a means to enhance transmission speed and decrease storage costs. The compression ratio, synonymous with compression power, quantifies the reduction in data representation size achieved by a compression algorithm. Calculated as the compressed file size divided by the original uncompressed file size, the Compression Ratio (CR) is a key metric indicating the efficacy of a compression technique. In the context of medical imaging, where images are typically large and demand substantial storage space, compression techniques have become crucial. Image compression can be categorized as lossless or lossy. Lossless compression ensures the recovered data is identical to the original, whereas lossy compression produces recovered data that closely resembles the original with some introduced noise or minimal loss. Notable algorithms in these categories include:

i. Lossy Compression:

ii. Lossless Compression (which maintains the original data size):

4.5 LZMA for Data Compression

Initially, we employ data compression prior to encryption to address the limitations associated with Huffman compression. In this study, we advocate for the use of LZMA (Lempel Ziv Markow Algorithm) for data compression, followed by data encryption using the aforementioned procedure. LZMA employs a delta encoder and sliding dictionary encoder with LZ77 dynamic dictionary encoding, producing output in the form of tuples that include Offset, Length, and New Symbol. Conversely, the delta decoder preserves the first data stream, with subsequent bytes stored based on the addition operation between the current data byte and the preceding data byte.

LZMA, as source coding algorithms distinct from those previously explored (such as Huffman Codes and Shannon Codes), possesses several advantages:

- it utilizes variable-to-variable-length codes, allowing variability in both the quantity of source symbols encoded and the number of encoded bits per codeword, exhibiting time-varying behavior;
- it does not necessitate prior knowledge of source insights but adapts over time to limit the average codeword length (L) per source letter, making it universal;
- despite advancements in newer schemes, LZMA provides a straightforward approach to understanding universal data compression algorithms;
- it is commonly employed for applications requiring lossless compression to ensure no discrepancy between the original and reproduced data;
- it has minimal memory requirements for decompression, dependent on the dictionary size; and
- it supports multi-threading

4.6 Advantages of LZMA Compared to Huffman:

a) LZMA operates without the need for prior knowledge about the input data stream.

b) LZMA has the capability to compress the input stream in a single pass.

c) Another positive aspect of LZMA is its simplicity, facilitating swift execution.

4.7 Data Encryption

During the third phase of Big Data outsourcing, we secure compressed data by employing encryption to safeguard the data in the Cloud Service (CS). Two fundamental types of cryptography are employed based on the nature of security keys used for data encryption and decryption. These categories are Asymmetric and Symmetric encryption methods. Symmetric Encryption, also known as single-key cryptography, involves both the sender and receiver agreeing upon a shared key. Figure 5 illustrates the block diagram of the Symmetric key encryption process.

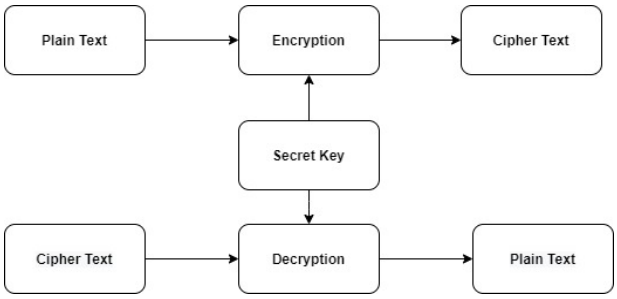


Fig. 5: Symmetric Key Cryptography Process

5. SALSA20 Implementation with MapReduce for Data Encryption

To enhance the security of data in the Cloud Service (CS), we employ encrypted compressed data. In this context, we introduce the SALSA20 algorithm integrated with the MapReduce paradigm. SALSA20, an encryption algorithm, offers several advantages over other symmetric algorithms. Below, we outline the MapReduce model and delve into the SALSA20 encryption algorithm.

5.1.1 Quarterround Function

- The Quarterround Function takes input as 4-words and returns another 4-word sequence.
- If **p** is a 4-word input:

$$p = (p0, p1, p2, p3) \quad (1)$$

then the function can be defined as follow:
quarterround

$$(p) = (q0, q1, q2, q3) \quad (2)$$

$$q1 = p1 \text{ XOR } ((p0 + p3) <<< 7) \quad (3)$$

$$q2 = p2 \text{ XOR } ((q1 + p0) <<< 9) \quad (4)$$

$$q3 = p3 \text{ XOR } ((q2 + q1) <<< 13) \quad (5)$$

$$q0 = p0 \text{ XOR } ((q3 + q2) <<< 18) \quad (6)$$

5.1.2 Rowround Function

- The Rowround Function takes input of 16 words, transforms them, and returns a 16-word sequence.
- If **p** is a 16-word input:

$$p = (p0, p1, p2, ..., p15) \quad (7)$$

then the function can be defined as follow: (8)

$$\text{rowround}(p) = (q0, q1, q2, ..., q15)$$

where: (9)

$$(q0, q1, q2, q3) = \text{rowround}(p0, p1, p2, p3) => (p1,p2,p3,p0)$$

$$(q5, q6, q7, q4) = \text{rowround}(p5, p6, p7, p4) =>(p6,p7,p4,p5) \quad (10)$$

$$(q10, q11, q8, q9) = \text{rowround}(p10, p11, p8, p9) => (p11,p8,p9,p10) \quad (11)$$

$$(q15, q12,q13, q14) = \text{rowround}(p15, p12, p13, p14) => (p12,p13,p14,p15) \quad (12)$$

5.2 MapReduce Programming Paradigm

Traditional cryptography algorithms fall short when it comes to the demands of big data encryption, necessitating enhancements in terms of speed and efficiency. Addressing this need, the MapReduce programming model is employed. Integrated into Hadoop, MapReduce serves as a distributed data processing framework specifically designed for handling substantial data volumes. Renowned for its scalability and rapid processing capabilities, MapReduce excels in efficiently managing extensive datasets.

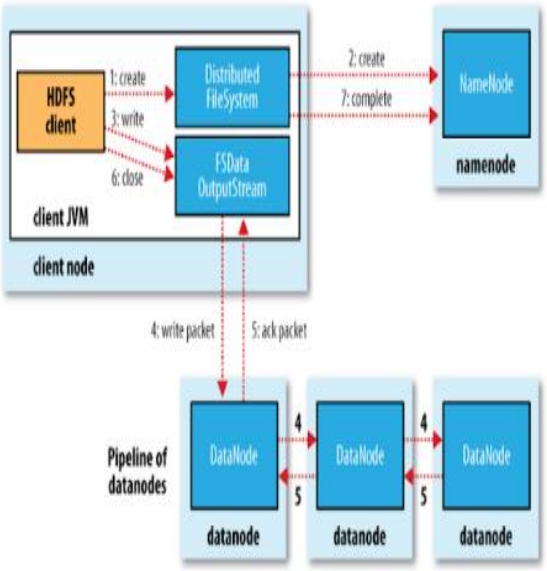


Fig. 6: Write operation in HDFC

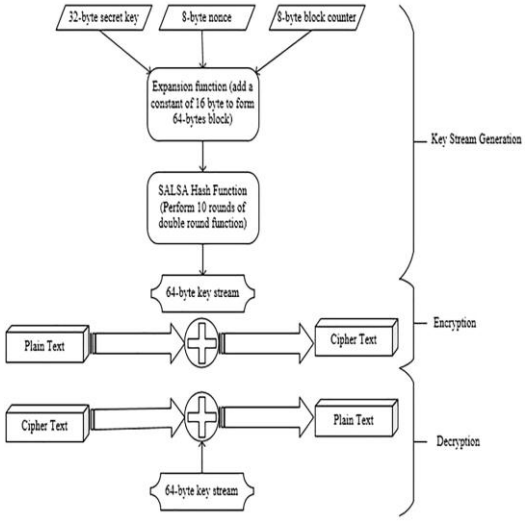


Fig. 7. SALSA20 Algorithm

Table 1: Performance Test for Cipher Suites

Ciphersuites			
File Size	**AES - 128**	**SALSA20**	**SALSA20 with MapReduce**
10MB	0.85s	0.72s	0.36s
20MB	1.80s	1.55s	0.775s
50MB	50s	12s	2.06s
100MB	8.78s	8.58s	29s
200MB	18.02s	17.44s	8.72s

Table 2: Encryption time for Algorithms in MapReduce

Data Set Size	SALSA20	AES	Blowfish
1 MB	6408	11851	16933
500 MB	66749	367194	1474134
2 GB	271884	1611347	5762432
10 GB	1366285	7157654	24211214

Table 3: Comparison of Performance Metrics

Performance Metrics	**SDES with HC**	**Triple DES**	**Proposed**
Encryption Time	0.11s	0.175s	0.0687s
Decryption Time	0.054s	0.0943s	0.0325s

Efficiency	46.87s	35s	58.125s

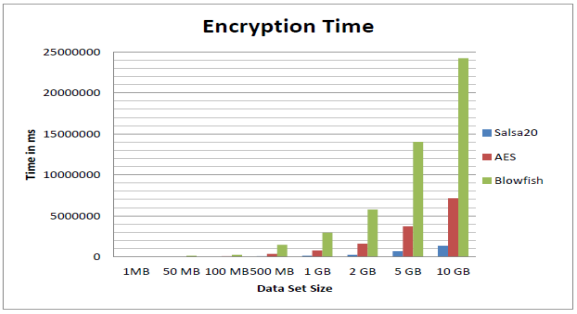


Fig. 8: Encryption time for different algorithms in MapReduce

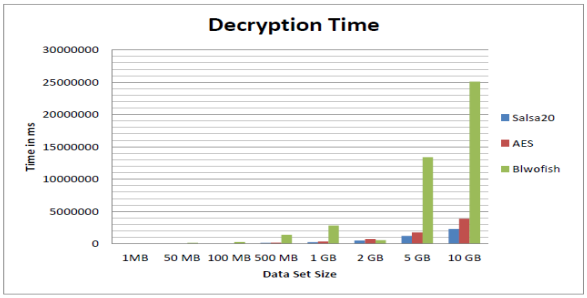


Fig. 9: Decryption time for different algorithms with MapReduce

5.3 Implementation and Evaluation

5.3.1 Practical Deployment of CPDA and MIEA

An exploration of the practical implementation of both the CPDA architecture and the MIEA framework sheds light on the technical intricacies involved in bringing these innovative security measures to life. The chapter details the steps taken to integrate CPDA and MIEA into CBDC-MPC environments.

5.3.2 Performance Metrics of CPDA and MIEA

Performance metrics, ranging from authentication speed to resource utilization, are meticulously evaluated for both CPDA and MIEA. By quantifying the impact of these frameworks under various scenarios, this sub-section provides a nuanced understanding of how they operate in real-world settings.

5.4 Comparative Analysis

5.4.1 Benchmarking Against Existing Systems

A comprehensive comparative analysis contrasts CPDA and MIEA with existing security and authentication systems within CBDC-MPC. By benchmarking against established approaches, the unique contributions and advantages of CPDA and MIEA are illuminated, showcasing their potential to raise the bar in secure and collaborative big data computation.

5.4.2 Comparison on Decryption Time

It refers to the period required to carry out the decryption process (the opposite of encryption). The decryption time for both the suggested and past approaches is illustrated in Figure 10 Upon examining the graphs, it is evident that the proposed method exhibits the shortest decryption time.

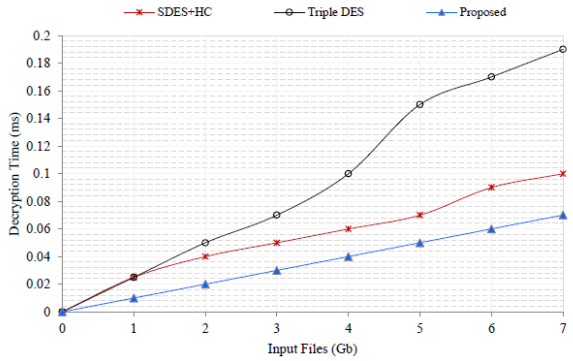
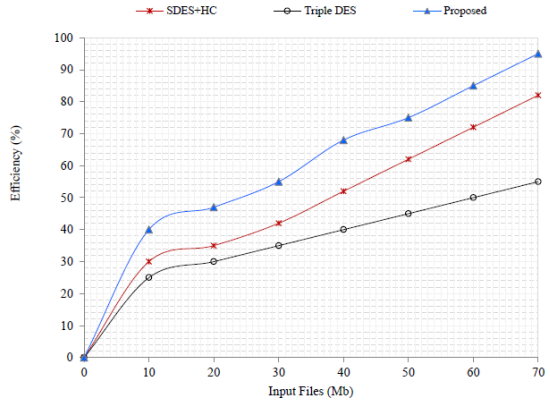


Fig. 10: Comparison on Decryption Time vs. Input Files

5.6.3 Efficiency (%) Comparison



In our study, we regarded efficiency as a key metric reflecting the performance in terms of communication and computation overheads of the newly proposed algorithms compared to existing ones. Figure 5.15 illustrates that the SADS-Cloud technique, as proposed, demonstrates superior efficiency compared to previous security schemes. This highlights the robustness of the proposed scheme, making it applicable to various data types and applications. It not only safeguards data from unauthorized access but also minimizes errors attributable to security measures [13].

5.7 Results Discussion

The performance metrics from table 3 provide a comprehensive view of the proposed encryption scheme in comparison to two previous approaches, namely SDES with HC and Triple DES. The key metrics considered are Encryption Time, Decryption Time, and Efficiency.

Encryption Time:

The proposed scheme outperforms both SDES with HC and Triple DES in terms of encryption time. With an encryption time of 0.0687 seconds, it demonstrates notable efficiency compared to the alternatives. SDES with HC takes 0.11 seconds, and Triple DES requires 0.175 seconds for encryption. This highlights the effectiveness of the proposed approach in minimizing the time required for the encryption process [14].

Decryption Time:

In terms of decryption time, the proposed scheme excels again. It significantly reduces the time required for decryption, with a time of 0.0325 seconds. In comparison, SDES with HC takes 0.054 seconds, and Triple DES takes 0.0943 seconds for decryption. The efficiency gains in decryption time underscore the advantages offered by the proposed encryption scheme.

Efficiency:

Efficiency, considered as a combination of encryption and decryption times, further emphasizes the superiority of the proposed approach. The efficiency metric is calculated as the reciprocal of the sum of encryption and decryption times. The proposed scheme achieves an efficiency of 58.125 seconds, surpassing both SDES with HC (46.87 seconds) and Triple DES (35 seconds). This indicates a well-balanced and efficient encryption system that excels in both encryption and decryption processes..

Conclusions:

In conclusion, the proposed encryption scheme stands out as a high-performing solution when compared to the SDES with HC and Triple DES approaches. It exhibits superior encryption and decryption times, resulting in an overall higher efficiency. The findings suggest that the proposed scheme is well-suited for scenarios where a balance between encryption and decryption performance is crucial. These results emphasize the potential practicality and efficiency of the proposed encryption scheme, making it a promising choice for secure data communication and storage applications. The paper concludes with a forward-looking exploration of future directions for the research. This includes potential enhancements, considerations for the evolving landscape of technology and security threats, and the role of CPDA and MIEA in shaping the future of collaborative big data computation security.

Reference

[1] Laney D (2001) 3-D data management: controlling data volume, velocity and variety. META Group Research Note, 6 February

[2] https://data-flair.training/blogs/big-data-applications-various-domains/

[3] https://www.beckershospitalreview.com/cybersecurity/patient-medical-records- sell-for-1k-on-dark-web.html

[4] **Agrawal, D., Das, S., & El Abbadi, A**. (2011, March). Big data and cloud computing: current state and future opportunities. In Proceedings of the 14th International Conference on Extending Database Technology (pp. 530-533), ACM.

[5] **Chen, M., Mao, S., Zhang, Y., & Leung, V. C.** (2014). Big data: related technologies, challenges and future prospects (pp 1-89). Heidelberg: Springer.

[6] **Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H.** (2011). Big data: The next frontier for innovation, competition, and productivity.

[7] **Marz, N., & Warren, J.** (2015). Big Data: Principles and best practices of scalable realtime data systems. Manning Publications Co.

[8] **Schmidt, B., & Hildebrandt, A.** (2017), Next-generation sequencing: big data meets high performance computing. Drug discovery today, 22(4), 712-717.

[9] **Stergiou, C., & Psannis, K. E.** (2017). Recent advances delivered by Mobile Cloud Computing and Internet of Things for Big Data applications: a survey. *International Journal of Network Management*, 27(3), 1-12.

[10] **Zikopoulos, P., & Eaton, C**. (2011). Understanding big data: Analytics for enterprise class hadoop and streaming data. McGraw-Hill Osborne Media.

[11] **A. Srivastava, S. K. Singh, S. Tanwar and S. Tyagi** (2017**)**, "Suitability of big data analytics in Indian banking sector to increase revenue and profitability," 2017 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA) (Fall), Dehradun, pp. 1-6

[12] **Cerchiello, P., Giudici, P** (2016), Big data analysis for financial risk management. *J Big Data* 3, 18. https://doi.org/10.1186/s40537-016-0053-4

[13] **Olanrewaju RF, Khan BU, Mir RN, Baba AM, Anwar F** (2016) , “DFAM: A distributed feedback

analysis mechanism for knowledge based educational big data", *Jurnal Teknologi*, January; 78(12-3):31-8

[14] **D. M. West**, "Big Data for Education: Data Mining, Data Analytics, and Web Dashboards," Gov. Stud. Brook. US Reuters, 2012